1. 参考文献（中英文）
2. 实验对比 增加对比试验



不用像这么多 但至少也要三四个吧

还有就是Loss值 收敛问题 明显就是发散的

1. 数据集评价标准，怎么正负面信息的
2. 具体程序算法过于简单

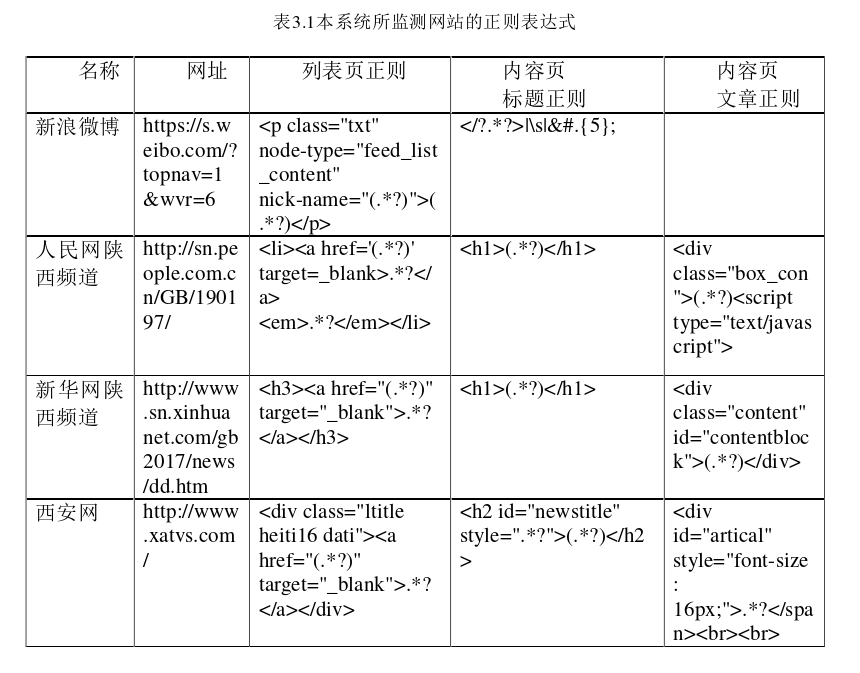
对算法 模型需要再做改进

1. 系统前端优化
2. 知识图谱是怎么可视化的
3. 数据库的搭建是怎么搭建的

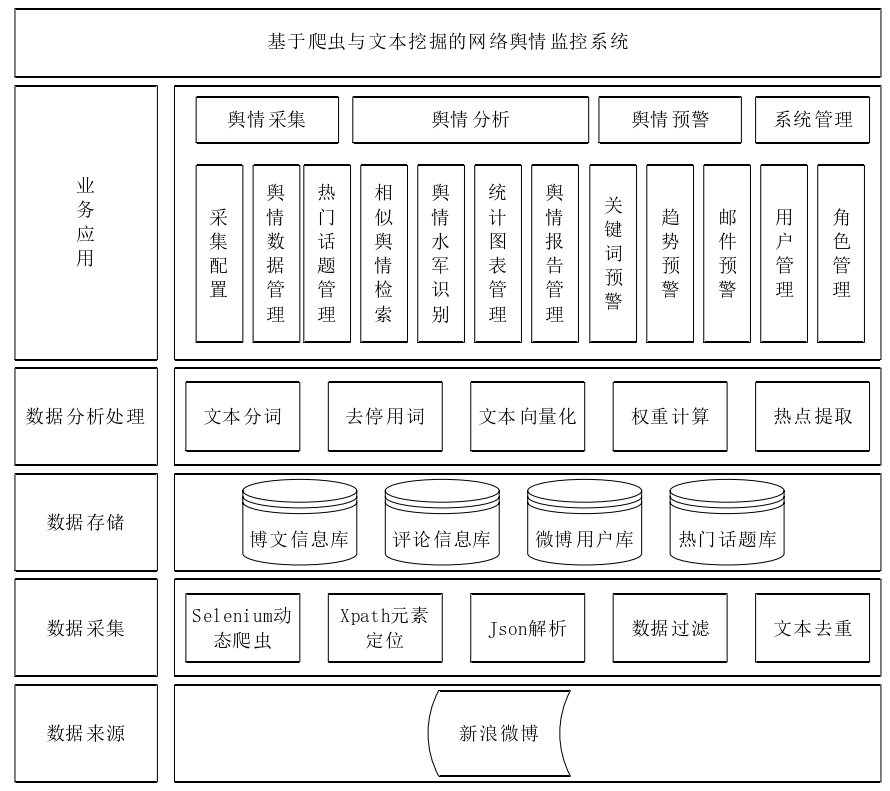
具体设计了哪些表 怎么设计的

1. 爬虫理论方面再具体化

正则表达式、数据来源



像这种的需要程序老师那边说明下

1. Transformer过度
2. 国内外研究现状扩充、系统性概括
3. 

还需要详细的列下这种系统架构图

12、需求分析、可行性分析

2，3，4，5，6，7，8

第2和4 是在一起的，我们弄到一起改进和修改试验，估计会增加一个更复杂的LSTM 或者 Bidirectional LSTM 和随机森林分类模型 至少增加2种新的分类器，加上原来的刚好3种（这个谈需求时候 我们只说了做下深度学习模型，没说需要做3种啊，就不算新需求），模型会更复杂，做sequence处理；

第5个我会优化下前端样式，进行美化

6，7，8 会在另外弄个文档说明」

**6，7，8问题文档说明部分下面👇：**

**第6个问题：知识图谱是怎么可视化的**

Answer：

实现知识图谱的具体步骤如下：

数据准备：首先需要准备知识图谱的数据，可以我们自己搜集的，填充到我们的web平台中。

前端界面设计：使用 Vue.js 搭建前端界面。可以使用 Element UI 或者 Vuetify 等组件库来构建前端 UI。

数据可视化：使用 D3.js 来实现数据的可视化，D3.js 可以帮助我们将数据转换成可视化图表，如力导向图、树状图等。

后端 API 实现：使用 Flask 搭建后端 API，提供数据接口供前端调用，可以使用 Flask-RESTful 来构建 RESTful API。

数据查询和过滤：前端可以使用 Vue.js 提供的组件和方法来实现数据的查询和过滤，可以使用组件库中的 Table、Select、Input 等组件来实现数据的交互和过滤。

数据存储和管理：可以使用json等数据文姐来存储和管理知识图谱的数据，可以使用 Flask 提供的 SQLALchemy ORM 工具来实现数据的持久化。

安全性和权限控制： Flask-Security 等工具来实现后端 API 的安全性和权限控制，可以使用 Vue.js 提供的路由守卫和权限控制来实现前端的安全性和权限控制。

实现知识图谱需要掌握前端技术（Vue.js、D3.js）、后端技术（Flask、ORM）以及数据存储和管理技术（sqlchemy），

能够对知识图谱的数据进行分析和处理，提高数据的质量和可用性。

**第7问题数据库的搭建是怎么搭建的**

**具体设计了哪些表 怎么设计的**

t SQLAlchemy 是一个 Python 的 SQL 工具包，它提供了一种基于 ORM (对象关系映射) 和 SQL 表达式的交互式查询接口。通过 SQLAlchemy，我们可以方便地操作关系型数据库，如 MySQL、SQLite、PostgreSQL 等。

假设我们要实现一个简单的可视化系统，其中需要涉及到两个数据表：

用户表（User Table）和博客表（Blogs Table）。

定义表结构

首先，我们需要使用 SQLAlchemy 定义数据表的结构。

需要使用 SQLAlchemy 的 ORM 功能来实现 Python 类和数据库表之间的映射关系。

对于用户表（User Table），我们可以定义一个名为 User 的 Python 类。该类需要继承自 SQLAlchemy 提供的基类 Model，并定义相应的属性（即字段）。

对于博客表（Blogs Table），我们可以定义一个名为 Blogs 的 Python 类。

该类需要继承自 Model 类，并定义相应的属性。

连接数据库

在定义好表结构之后，我们需要使用 SQLAlchemy 连接数据库。具体而言，需要指定数据库类型（如 MySQL、SQLite、PostgreSQL 等）、数据库地址、数据库端口号等信息，以及数据库用户名和密码等认证信息。

创建数据表

在连接好数据库之后，我们需要使用 SQLAlchemy 自动创建数据表。具体而言，需要使用 SQLAlchemy 的 create\_all() 方法创建数据表。

数据操作

创建好数据表之后，我们就可以使用 SQLAlchemy 进行数据操作了。具体而言，我们可以通过 SQLAlchemy 提供的查询接口来查询数据，通过添加、修改、删除接口来操作数据。

SQLAlchemy 实现用户表和博客表的基本步骤。需要注意的是，上述步骤涉及的具体细节因数据库类型而异，需要进行相应的调整

**8 爬虫正则表达式**

对于 "https://china.huanqiu.com/",

<https://mil.huanqiu.com/>

**2个站点的正则表达式：**

| **正则表达式** | **描述** |
| --- | --- |
| **//ul[@id="recommend"]//a** | 匹配新闻列表中的所有新闻链接 |
| **.//h4/text()** | 匹配新闻标题 |
| **./@href** | 匹配新闻链接 |
| **//section[@data-type="rtext"]/p** | 匹配新闻内容中的所有段落 |
| **//div[@class="metadata-info"]//p[@class="time"]** | 匹配新闻发布时间 |
| **//div[@class="metadata-info"]//span[@class="source"]//a** | 匹配新闻来源链接 |
| **//div[@class="metadata-info"]//span[@class="source"]//span** | 匹配新闻来源名称 |

**对于新浪本地新闻**：

抓取规则是：

使用BeautifulSoup和requests库进行网页请求和解析

对于新闻列表页面，通过请求和解析获取新闻的标题和链接

对于每个新闻链接，再次请求和解析获取新闻的详细信息，包括标题、发布时间、来源、作者、正文内容和链接

将获取的新闻标题和正文内容写入到本地文本文件中，并存入数据库中（如果数量未超过设定的限制）